Introduction to Random Numbers, Sampling, and MCMC Methods

Bill Watson

S&P Global

August 7, 2019

What is Sampling and why is it useful?

- Sampling is the practice of generating observations from a population
- Monte Carlo methods are algorithms that rely on repeated random sampling to obtain approximations where it is difficult or impossible to use deterministic approaches
    - Optimization
    - Numerical Integration
    - Sampling distributions

Application: Approximating π

Algorithm 1 Approximating π

Input: Batch Size N

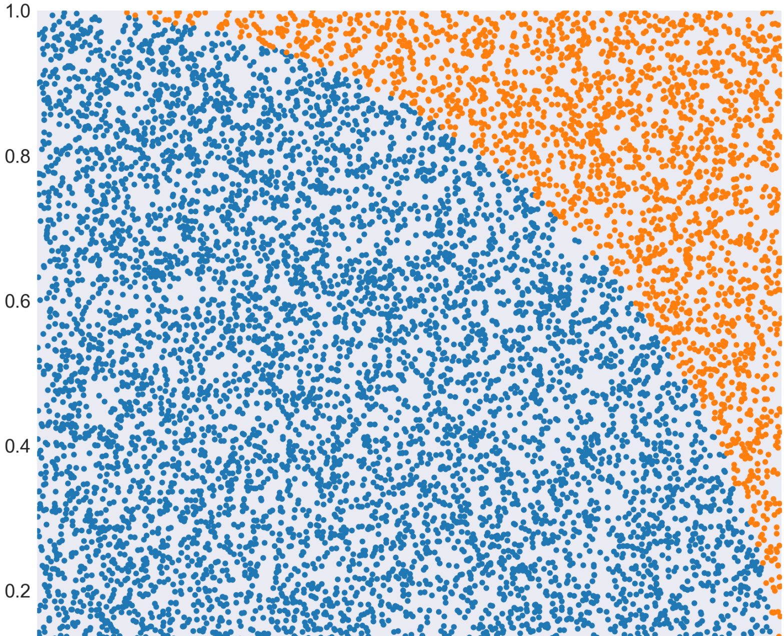
1: Sample $u_1 \sim \text{Uniform}(0,1)$ N times

2: Sample $u_2 \sim \text{Uniform}(0,1)$ N times

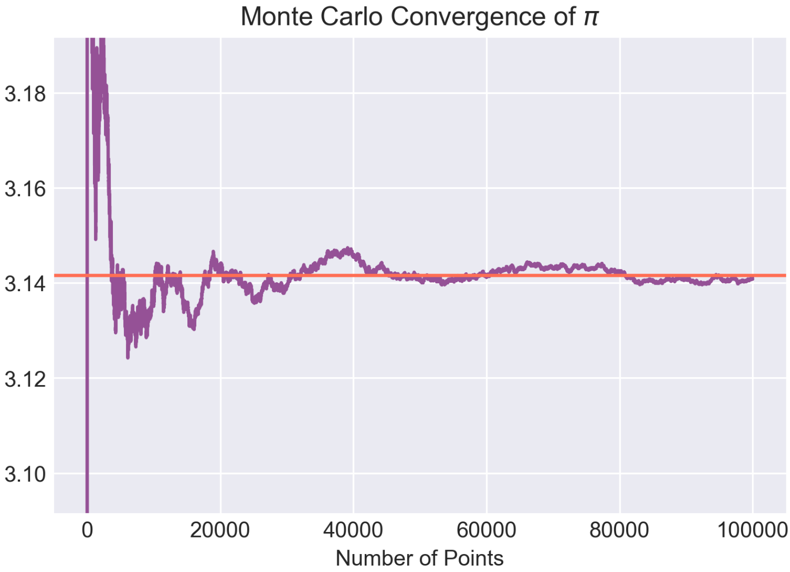
3: $\tilde{\pi} = \frac{4}{N} \cdot \left| \left\{ (u_1, u_2) \;\middle|\; \sqrt{u_1^2 + u_2^2} < 1 \right\} \right|$

Output: $\tilde{\pi}$

Application: Approximating π



Application: Approximating π



What can we do with our samples?

- Integration: $\int p(x)\, f(x)\, dx = \frac{1}{n} \sum_{x_i \sim P} f(x_i)$
- Expectation: $\mu = \frac{1}{n} \sum_{x_i} x_i$
- Variance: $\sigma^2 = \frac{1}{n} \sum_{x_i} (x_i - \mu)^2$
- Median: $\text{median} = \text{median}(x_1, x_2, \ldots x_n)$
- Entropy: $\mathbb{H}(P) = -\frac{1}{n} \sum_{x_i \sim P} \log p(x_i)$
- CDF: $p(c) = \frac{1}{n} |\{x_i \mid x_i \leq c\}|$

Pseudo-Random Number Generators

Pseudo-Random Number Generators: LCG

Algorithm 2 Linear Congruential Generator

Input: Modulus m, Multiplier a, Increment c, Seed X_0

1: $X_{i+1} = (a \cdot X_i + c) \mod m$

Output: X_{i+1}

Pseudo-Random Number Generators: Mersenne Twister

Generating a Normal from the CDF

- We can use the cumulative distribution function to sample any distribution
- For instance, a normal's CDF is:

$$F(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$$

- With an Inverse CDF as:

$$F^{-1}(p) = \mu + \sigma\sqrt{2} \cdot \operatorname{erf}^{-1}(2p-1)$$

- $\operatorname{erf}(x)$ is the error function, defined as:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

Inverse Transform Sampling

- It's easy to generalize this method to any distribution with a closed-form inverse CDF

Algorithm 3 Inverse Transform Sampling

Input: Inverse CDF F^{-1}

1: Sample $u \sim \text{Uniform}(0, 1)$

2: $X = F^{-1}(u)$

Output: X

Inverse Transform Sampling: Example

Table of Inverse CDFs for Common Distributions

Distribution	$F^{-1}(p)$
$\mathcal{N}(\mu, \sigma)$	$\mu + \sigma\sqrt{2} \cdot \mathrm{erf}^{-1}(2p - 1)$
$\mathrm{Uniform}(a, b)$	$a + p \cdot (b - a)$
$\mathrm{Exp}(\lambda)$	$\frac{-\ln(1-p)}{\lambda}$

Rejection Sampling

Rejection Sampling: Algorithm

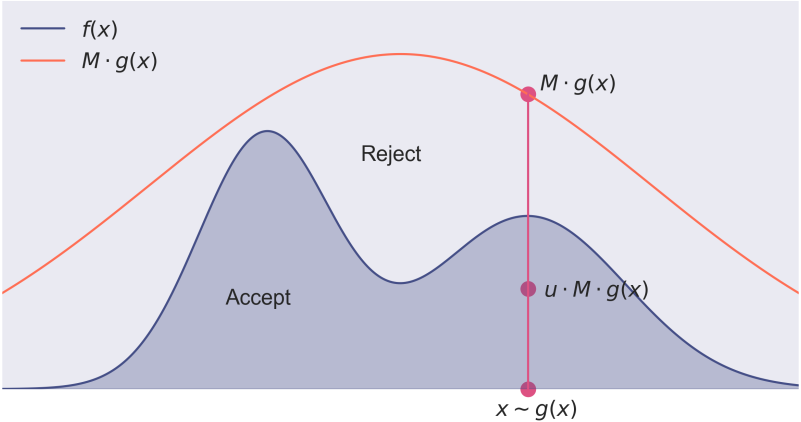
Algorithm 4 Rejection Sampling

Input: Model P, Proposal Q, $M > 1$

```
Sample x ~ P
Sample u ~ Uniform(0, 1)
if u < p(x) / (M·q(x)) then
    Accept x
else
    Reject x
end if
```

Output: Accepted Samples

Rejection Sampling: Intuition



Rejection Sampling: Example

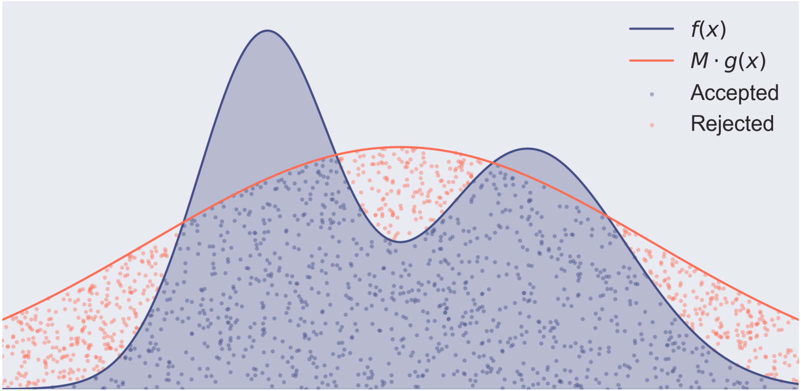


Figure: Rejection Sampling with $M = 1.3$

Rejection Sampling: Example

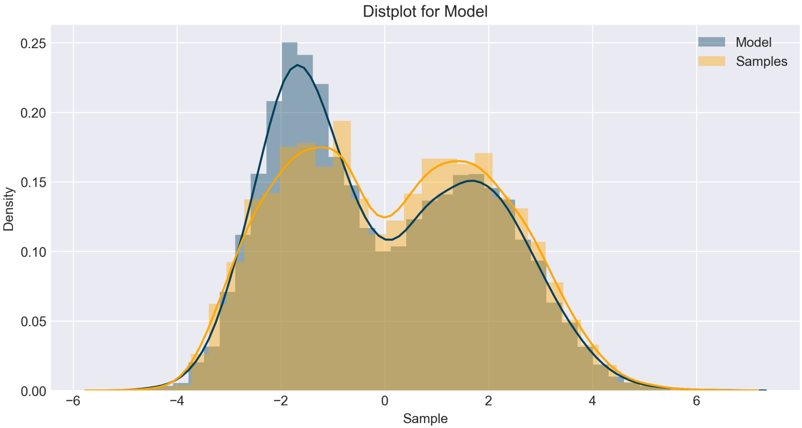


Figure: $M = 1.3$, $6,688$ Accepted Samples

Rejection Sampling: Example

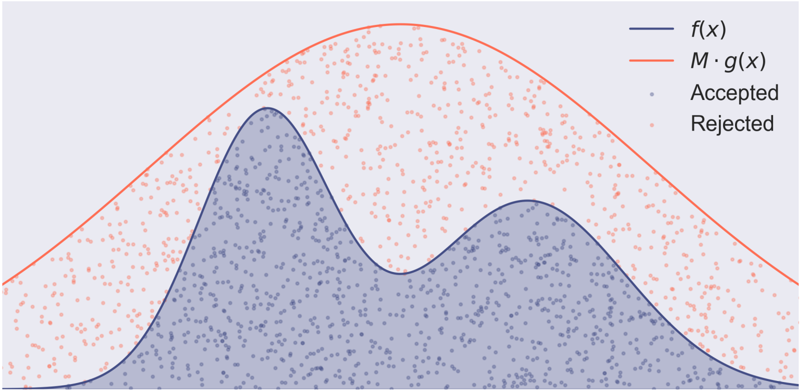


Figure: Rejection Sampling with $M = 2.5$

Rejection Sampling: Example

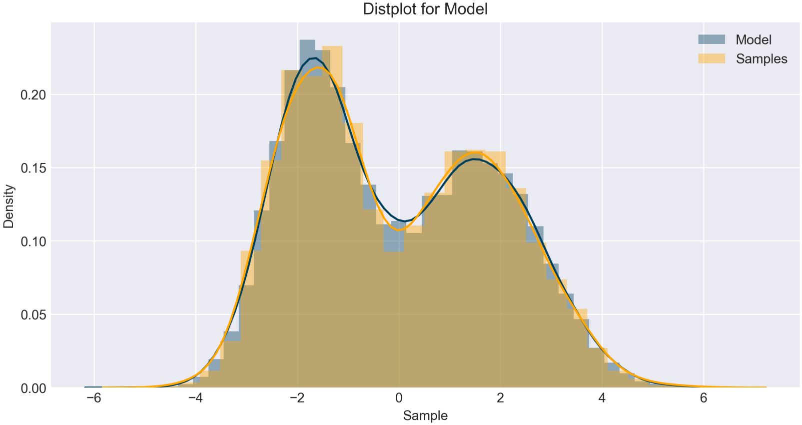


Figure: $M = 2.5$, $4,085$ Accepted Samples

Rejection Sampling: Example

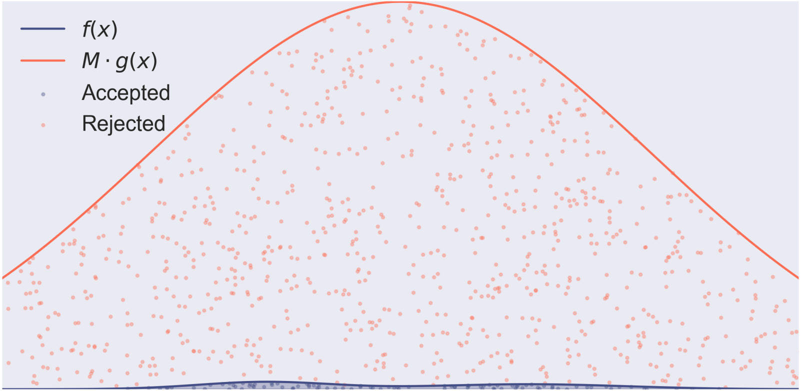


Figure: Rejection Sampling with $M = 100$

Rejection Sampling: Example

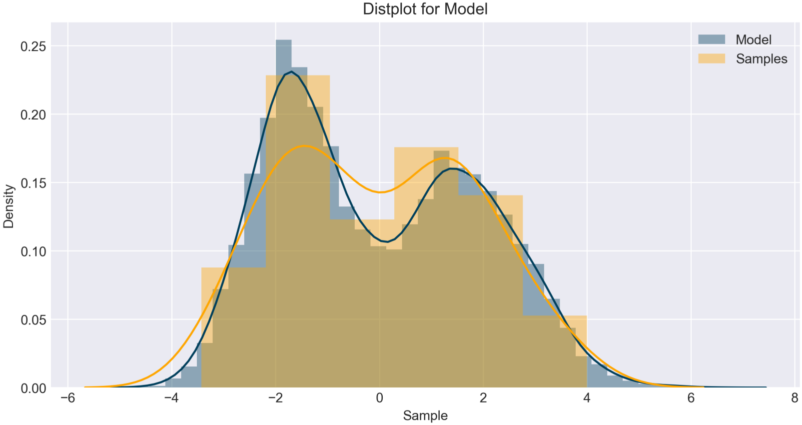


Figure: $M = 100$, 92 Accepted Samples

Rejection Sampling: Pros & Cons

What is MCMC?

Metropolis-Hastings

Metropolis-Hastings: Algorithm

Algorithm 5 Metropolis-Hastings Algorithm

Input: Model P, Proposal Q

Initialize x_0
for $s = 0, 1, \ldots$ **do**
    Sample $x' \sim q(x'|x_s)$
    Compute acceptance probability

$$r = \min\left(1,\ \frac{p(x')}{p(x_s)}\frac{q(x_s|x')}{q(x'|x_s)}\right)$$

    Sample $u \sim \text{Uniform}(0, 1)$
    if $u < r$ **then**
        Accept x'
        Set $x_{s+1} = x'$
    else
        Reject x'
        Set $x_{s+1} = x_s$
    end if
end for

Output: Accepted Samples

Metropolis-Hastings: Example

Metropolis-Hastings: Conditions

Variants of Metropolis-Hastings

- Metropolis Algorithm
  - Uses a symmetric proposal distribution Q, such that: $q(x_s|x') = q(x'|x_s)$
- Metropolis-Adjusted Langevin Algorithm
  - New states are proposed with Langevin dynamics: $x' = x_s + \tau\nabla\log\pi(x_s) + \sqrt{2\tau}\xi_k$
  - Proposal probabilties are normally distributed: $q(x'|x_s) \sim \mathcal{N}(x_s + \tau\nabla\log\pi(x_s), 2\tau I_d)$
- Hamiltonian Monte-Carlo
  - Adds Hamiltonian Dynamics
- No-U-Turn Sampler (NUTS)
  -

Gibbs Sampling

Further Reading

- Machine Learning: A Probabilistic Perspective by Kevin Murphy
- Monte Carlo Methods in Financial Engineering by Paul Glasserman
- Probabilistic Graphical Models: Principles and Techniques by Daphne Koller and Nir Friedman